Sort-Select-Damp: An efficient strategy for color look-up table lattice design

Vishal Monga and Raja Bala, Xerox Research Center Webster, NY, USA

Abstract

Real-time processing constraints entail that non-linear color transforms be implemented using multi-dimensional look-up-tables (LUT). The LUT cannot be prohibitively large because of storage and memory constraints. Hence the LUT is built with a sparser input node sampling. The issue of LUT node placement becomes important in such cases. The standard approach is to place the nodes upon a uniform regular lattice spanning the entire input color space. Such a uniform placement of LUT nodes ignores certain crucial factors namely the curvature of the non-linear color transform, and the statistical distribution of the input – often resulting in high approximation errors or equivalently objectionable visual artifacts in images processed through the LUT. In this paper, we formulate the underlying cost measure that node placement algorithms should seek to minimize. At the heart of this measure is a significance function which quantifies the relative importance of input variables. We argue that techniques which select truly optimal nodes w.r.t this cost measure are computationally infeasible. We then propose an efficient algorithm that is essentially based on selecting nodes that lie at maxima of the significance function. The latter is iteratively adjusted to prevent degenerate node selection. Experimental results across a variety of scenarios demonstrate significant transform accuracy improvements over classical uniform node-spacing.

1. Introduction

In many instances along an image path, it is necessary to apply color transformations to images and objects. Often, these transformations are complex non-linear functions making it impractical to process large images in real-time. It is therefore common to implement such mathematical transforms as a multi-dimensional look-up-table. The look-up-table cannot be prohibitively large because of processor RAM and cache memory constraints. Hence the look-up-table is built with a sparser input node sampling. The issue of "node placement" becomes important in such cases.

The standard approach is to place nodes upon a uniform regular lattice [1]. Non-uniform node spacing on a lattice offers benefits in scenarios where the multi-dimensional transform to be realized via the LUT exhibits varying curvature as a function of the input variables to the LUT. Much attention has been given to adapting to the transform both in terms of practical algorithms [2, 3], and theoretical results in asymptotic cases where the number of lattice nodes becomes large [2]. The existing algorithms for node selection in multidimensional LUTs [2, 3] however are either computationally burdensome or based on a sequential structure that proceeds as separable in each dimension [2]. There is a two-fold motivation then for the work presented in this paper: 1.) there is a need for *efficient* algorithms that can select *truly* multi-dimensional nodes, and 2.) while transform curvature is important in many color imaging applications, the distribution of input image colors, or more generally, a weighting function that quantifies relative importance of image colors, can also be a crucial factor in node selection for color LUTs. A good example is the use of a LUT to implement color editing functions tailored to a specific image.

This paper proposes an algorithm that accounts for both these considerations in node selection for LUT lattices. We formulate a cost measure that node selection algorithms must seek to optimize (minimize), and subsequently argue that algorithms must resort to combinatorial search to obtain optimal nodes. We hence propose an efficient, albeit sub-optimal algorithm that adapts or selects LUT nodes based on a significance function of input variables. This significance function allows us to unify the concerns of input as well as transform adaptability. Our algorithm has modest complexity, i.e. order M, where a M^m LUT is desired, m being the dimensionality of the input variable to the LUT. A bonus outcome of our algorithm is that it can automatically make recommendations for a satisfactory LUT size based on the "significance function".

The rest of the paper is organized as follows. Section 2 sets the notation for the paper, and formulates the cost measure or equivalently the optimization problem of interest. Section 3 presents a new sub-optimal node selection scheme that accounts for critical elements of the cost measure such as the input distribution and the transform function curvature. Experimental results are presented in Section 4. Section 5 concludes the paper with suggestions for future work.

2. Problem Formulation

Let $\mathbf{x}$ in S, where S is a subset of R^m denote the multi-dimensional input variable to the look-up-table. Also, let g ( ): $R^m \rightarrow R^n$ represent the multi-dimensional transform that the LUT is designed to represent. In a typical color transform implementation, examples of the LUT input variable $\mathbf{x}$ and corresponding output value g($\mathbf{x}$) include RGB, Lab, CMYK etc. Because the LUT realization only provides an approximation to the true transform, we define the approximation error as

$$E(S_N) = \int_{\mathbf{x} \in S} p(\mathbf{x}) \left\| g(\mathbf{x}) - \hat{g}(\mathbf{x}, S_N) \right\|^2 d\mathbf{x} \qquad (1)$$

Where:

S_N (a subset of S) $=\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ represents the set of nodes or equivalently the lattice that defines the LUT.

$\hat{g}(\mathbf{x}, S_N)$ – is an approximation to g( ) built out of evaluations of g( ) at the node points and suitable multi-dimensional interpolation for all **x** in between.

$p(\mathbf{x})$: $R^m \rightarrow R^+$, a non-negative input importance function. Example instantiations include a histogram of input colors to the LUT, or a map that attaches greater importance to memory colors such as skin tones, etc.

The cost measure $E(S_N)$ in Eqn. (1) is clearly a function of the LUT nodes. The goal of LUT design algorithms is hence *to make an intelligent choice of* S_N. To that end, it must be realized that for a fixed number of nodes N, an optimal selection of the subset S_N of S that globally minimizes $E(S_N)$ is a hard problem. Part of the fundamental difficulty results from the fact that the inputs are multi-dimensional. Sequential approaches that use separable grid structures [2] have been investigated with some success for providing efficient schemes. That said, methods that select truly multidimensional nodes typically do not scale well as a function of the input dimension *m* [3].

The discerning reader may also observe that the solution to Eqn. (1) depends on several factors like the multi-dimensional transform g( ), the dimensionality *m* of the input and type of interpolation used, i.e. linear, popular 3-D, 4-D geometries like tetrahedral, pentahedral etc. An appreciation of the hardness of the problem can also be gauged by observing the special case, $g(\mathbf{x}) = \mathbf{x}$, and $g(\mathbf{x}, S_N) = Q(x)$, i.e. vector quantization of x to the set S_N; which is well studied [4], and best known algorithms can only provide local minima. In fact, without loss of generality, it can shown by equivalent problems in graph theory [5] that the problem in Eqn. (1) lies in the NP-class, and finding the optima entails computationally infeasible combinatorial search.

3. Sort-Select-Damp: An efficient, sub-optimal approach

In this work, we present an algorithm that provides a simple and efficient solution to the node selection problem in Eqn (1). While we do not make guarantees of optimality, our algorithm incorporates the key components of the cost measure, namely the input distribution $p(\mathbf{x})$, and the curvature of the multi-dimensional transform $g(\mathbf{x})$.

The inputs to our algorithm comprise:

a. A set of input values $S = \{\mathbf{x}_i \in R^m, i = 1,2,..n\}$ which serves as a statistical representation of the source/input data.

b. A non-negative input importance function $p(\mathbf{x})$ as described earlier in Section 2.

c. A distance function $d(\mathbf{x},\mathbf{y})$ such that:

$$d(\mathbf{x},\mathbf{y}) >= 0, \text{ for all } \mathbf{x}, \mathbf{y} \text{ in S.} \quad (2)$$
$$d(\mathbf{x},\mathbf{y}) = 0, \text{ implies } \mathbf{x} = \mathbf{y} \text{ and vice-versa.}$$

The algorithm is then divided in three parts:

1) Preprocessing to define a significance function that brings together the factors of input image distribution and function curvature into LUT node design.

2) Multi-dimensional node selection via *sort-select-damp* iterations, and

3)Post-processing: separable per-dimension operation to enable compliance with pre-determined rules of number of node levels and node spacing along each individual dimension.

Each step is now elaborated.

Step 1: Preprocessing

Define a non-negative significance function $s(\mathbf{x})$ as follows:

$$s(\mathbf{x}) = \alpha.p(\mathbf{x}) + (1-\alpha).\left|\det(Hessian(g(\mathbf{x})))\right| \quad (3)$$

The first term in the linear weighting is the input distribution. The second term uses the magnitude of the determinant of the second derivative matrix (Hessian) of $g(\mathbf{x})$ as a measure of the transform curvature. The quantity α in [0,1] trades off relative importance of image distribution vs. function curvature. Unless, stated otherwise it is also assumed for ensuing discussion that $s(\mathbf{x})$ is normalized to lie in [0, 1].

Step 2: Multi-dimensional node selection based on "Sort-Select-Damp" procedure

Fig. 1 Illustrates this procedure.

- The "sort" step simply arranges the nodes in decreasing order of significance.
- The "select" step picks the node(s) with the maximum significance.
- The significance function is then multiplied by a damping function that ensures that no subsequent node selections lie too close to the current selection of nodes. $\mathbf{n}_j$ is an m-dimensional node value, with corresponding values $\mathbf{n}_j(k)$ along the k-th dimension.

Several choices of the distance or dampening function $d(\mathbf{x}, \mathbf{y})$ are possible. The most general requirements were given in Eqn. (2). We'll illustrate shortly in Fig. 2 that the distance function essentially trades-off between the significance function $s(\mathbf{x})$ and the relative positioning of the nodes. It is therefore important to select the distance function carefully so that an appropriate balance is obtained. Inspired by previous efforts in color palette design [6], we pick a distance function of the form:

$$d(\mathbf{x},\mathbf{y}) = (1 - e^{-\beta\|\mathbf{x}-\mathbf{y}\|^2}), \beta > 0 \quad (4)$$

The damp step is crucial in that it re-assigns values to the significance function $s(\mathbf{x})$ by "dampening" it with a distance weighting function $d(\mathbf{x}, \mathbf{y})$. Such dampening serves two purposes: 1.) for $\mathbf{x} = \mathbf{n}_j$ the revised $s(\mathbf{x})$ is forced to 0 making sure that it is not

selected again, and 2.) a distance penalty is attached for picking nodes "too close" to each other.

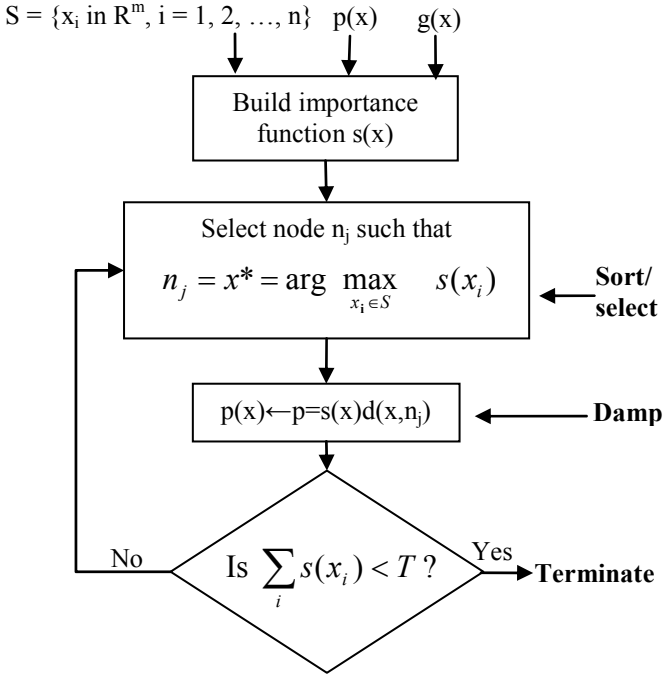
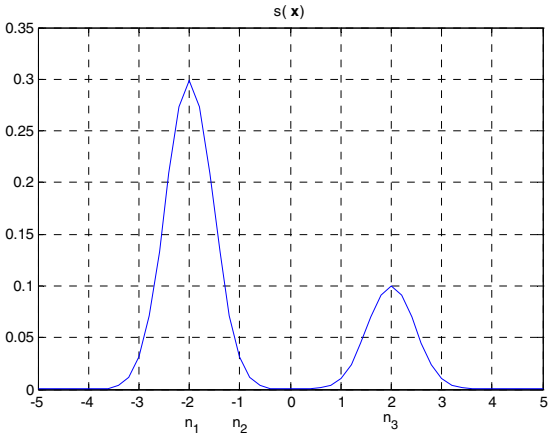
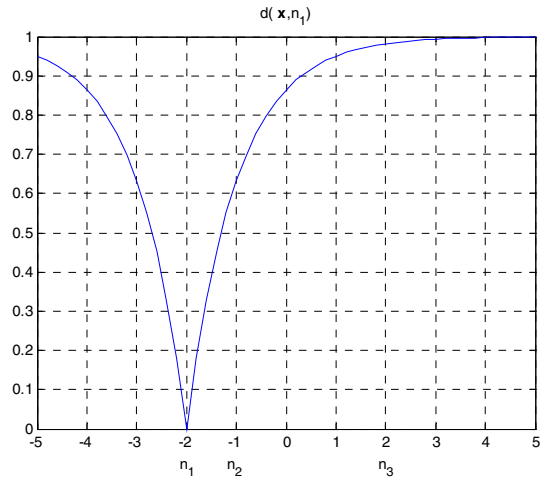


Fig. 1 Node selection via sort-select-damp procedure

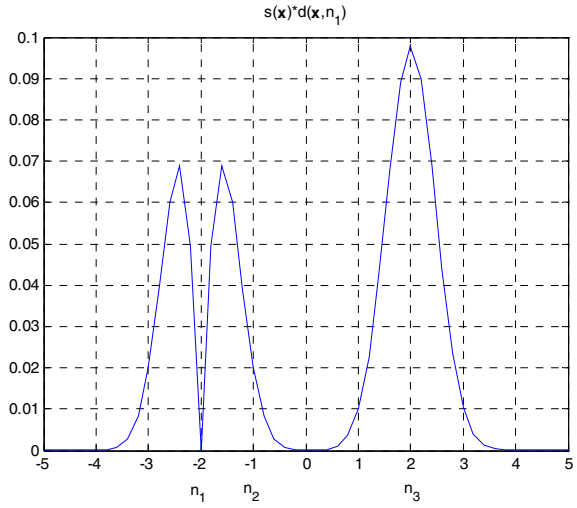
The second point is illustrated below in Fig. 2 by a 1-dimensional example. Fig. 2(a) plots s(**x**) vs. **x**, Fig. 2(b) plots d(**x, n**$_1$) where n$_1$ denotes the node that is selected first, i.e. n$_1$ = arg max s(**x**). In Fig. 2(c) we plot s(**x**).d(**x, n**$_1$). The significance of the dampening function d(.,.) is readily apparent from Figs. 2 (a)-(c). In particular, in the absence of such a distance based dampening of s(**x**), all nodes would be selected in very close proximity of n$_1$ resulting in undesirable node spacing.



(a)



(b)



(c)

Fig. 2: Illustrating the effect of the dampening function

Note that the termination criterion is directly related to the area under the distance-weighted significance function s(**x**). This quantity can be thought of as a rough indication of the approximation error of the LUT. Thus for a given s(**x**) and distance weighting d(**x,y**), the threshold T automatically determines the number of nodes selected through the process, i.e. smaller T results in a greater number of nodes. A reasonable value of T can be determined empirically by experimenting over many data sets

Step 3: Post-processing algorithm

One potential problem with the aforementioned algorithm is that while the nodes **n**$_j$ are well separated in m-dimensional space, the projections onto the individual LUT axes might result in node levels being coincident or excessively close to each other. We thus perform a post-processing step to avoid this situation and enforce a minimum distance between adjacent nodes along each input dimension of the LUT. Additionally, for some applications it may be necessary to impose constraints on the number of node levels along each dimension. For example, ICC profiles require the same

number of node levels along each dimension. To satisfy this constraint, additional node levels may have to be added along some of the dimensions. This is accomplished in a subsequent post-processing step. Let us denote the number of multi-dimensional nodes $\{\mathbf{n}_j\}$ that the algorithm terminates with as M, i.e. j = 1,2,…M. Without loss of generality, let $\{n_1, n_2, n_M\}$ represent *sorted scalar* values of the nodes in the k^{th} dimension. That is,

$$\{n_1, n_2, ..., n_M\} = \text{sort}(\mathbf{n}_1(k), \mathbf{n}_2(k), ..., \mathbf{n}_M(k))$$

And hence, $n_1 < n_2 < \ldots < n_M$.

Let *delmin* represent the minimum allowable node-spacing along the k^{th} dimension. Further, let n_{min} and n_{max} represent the "boundary values" that must be appended to the node vector. In particular, $n_{min} < n_1$ and $n_{max} > n_M$. Also denote $\mathbf{s}_k(n)$ to be the separable significance function for the k-th dimension obtained from the joint multi-dimensional significance function s(**x**). As before, $\mathbf{s}_k(n)$ is normalized to lie in [0, 1]. To make sure that the boundary values always appear in the final node selection we set $s_k(n_{min}) = 1$, and $s_k(n_{max}) = 1$. Also let us represent n_{min} by n_0 and n_{max} by n_{M+1}.

Step 3A: Enforcing priority-based minimum node-spacing

if ($(n_{i+1} - n_i) < delmin$)
  if $s(n_{i+1}) > s(n_i)$ then retain n_{i+1} and eliminate n_i
  else retain n_i and eliminate n_{i+1}

Decrement i till end of node list. Repeat for all consecutive node pairs in the list.

Clearly the above procedure cannot increase the number of nodes in the list. Hence assume after this step we are left with K nodes along the k^{th} dimension, where K < M+2.

Step 3B: Node addition via median-cut placement

For reasons given earlier, it is sometimes necessary to enforce explicit constraints on the number of nodes along each dimension. An algorithm for accomplishing this is described next.

Let P_k denote the desired LUT size along the k^{th} dimension, k = 1,2,..m, where $P_k >= M$. As an example, a 3-D input LUT would make a lattice of size $P_1 \times P_2 \times P_3$. Given nodes n_1 through n_K:

While ($K < P_k$)
- Determine index i such that i is in 1,2,…K and that :
  $i = \arg\max |n_{i+1} - n_i|$, $i = 1, 2, ., N$
- Place a node $n^* = (n_i + n_{i+1})/2$ between n_i and n_{i+1}
- K → K + 1

The post-processing steps 3A and 3B are performed separately along each dimension to obtain a final lattice of size $N = \Pi P_i$.

4. Experimental Results

We present results for adapting the nodes of a 2-D LUT meant to approximate an $\mathbf{R}^2 \rightarrow \mathbf{R}$ function. The function to be approximated via the LUT is plotted in Fig. 3 (a). Additionally, the input to the function is designed to be drawn from a 2-D probability density function synthesized as the mixture of four Gaussians with means (10, 10), (10,-10), (-10,-10) and (-10, 10) respectively. The probability density is plotted in Fig. 3 (b). Fig. 3 (c) plots the magnitude of the determinant of the second derivative matrix, i.e. the Hessian for this function. A high value for the Hessian indicates regions of high curvature [3].

4.1 Visualization of Node Placement

We first intuitively demonstrate the merits of our algorithm by visualizing node-placement for three cases: 1.) density based placement, i.e. s(**x**) = p(**x**) or Eqn. (3) with = 1, 2.) curvature based placement or s(**x**) = | det(Hessian(g(x))) | or Eqn. (3) with = 0, and 3.) joint node placement by using an intermediate .

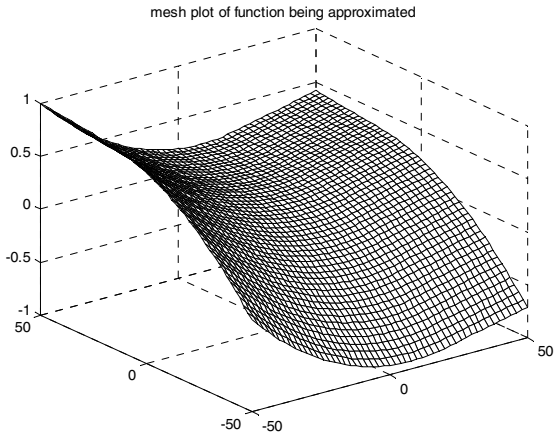
From Fig. 3 (b) it is clear that the density is highest in the neighborhood of the four local maxima (10,10), (10,-10), (-10,-10) and (-10,10). It is desirable hence that more LUT nodes should be placed in regions of higher density.

In Fig. 4 (a), a uniform 11 x 11 grid is overlaid on the contours of the probability density in Fig. 3 (b), while in Fig. 4 (b), the same contour plot is shown but this time overlaid with 2-D LUT nodes derived using our proposed algorithm, and density based placement. Comparing Figs. 4 (a) and 4 (b), it is clear that the proposed LUT node selection respects the underlying input density while the uniform grid does not. Fig. 4 (c) shows node placement based on setting the significance function to simply a measure of curvature. Comparing Figs. 4 (c) and 3 (c), it is clear that once again the node placement is dense in regions of high curvature and sparse otherwise. Finally, Fig. 4 (d) visualizes node placement when a linear combination of the density and curvature measures instantiated with = 0.3 was used as the significance function for our algorithm. As Fig. 4 (d) reveals, in this case, the node placement respects both considerations. In practice, could be optimized w.r.t an ensemble of representative input images and color transformations.

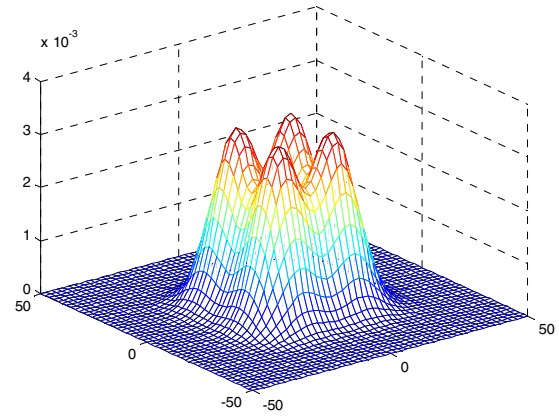
4.2 Quantitative Measure of Function Approximation

Next, we provide quantitative results in showing improvements in approximating the 2-D transform via the LUT designed using our proposed algorithm.

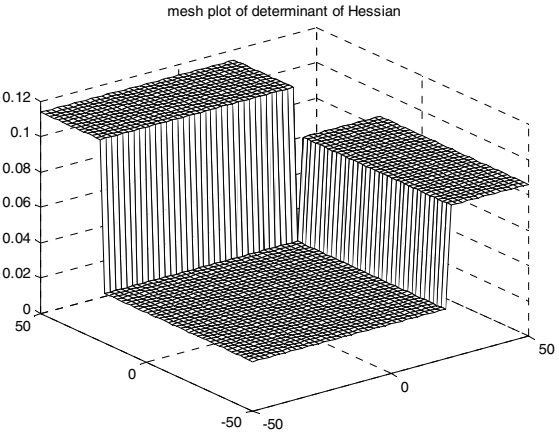
Recall the error function $E(S_N)$ in Eqn. (1) evaluated for a certain node placement. To compute this error with generated 1000 2-D input values to be processed through the LUT from the Gaussian mixture density described earlier. Bilinear interpolation was used to determine outputs for input values other than the LUT nodes. The gain achieved by using proposed lattice design that adapts to a significance function can then be expressed as:



(a)



(b)



(c)

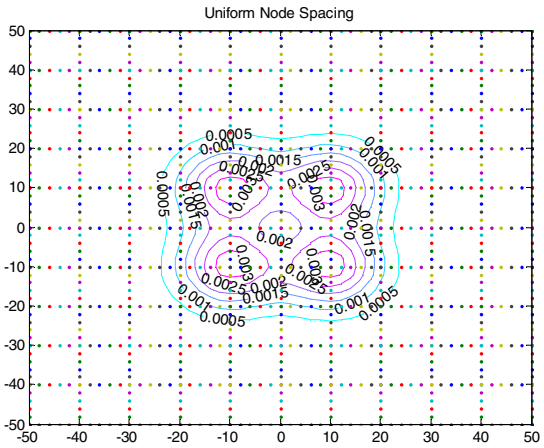
Fig. 3: (a) the 2-D function to be approximated, (b) the 2-D probability density function of input variables, (c) plot of the magnitude of the Hessian of the 2-D function, curvature is high where the Hessian peaks.

LUT Lattice Design	**Gain in dB over uniform node spacing**
Probability Density Based ($\alpha = 1$)	18 dB
Curvature Based ($\alpha = 0$)	10.3 dB
Based on joint significance function ($\alpha = 0.3$)	21 dB

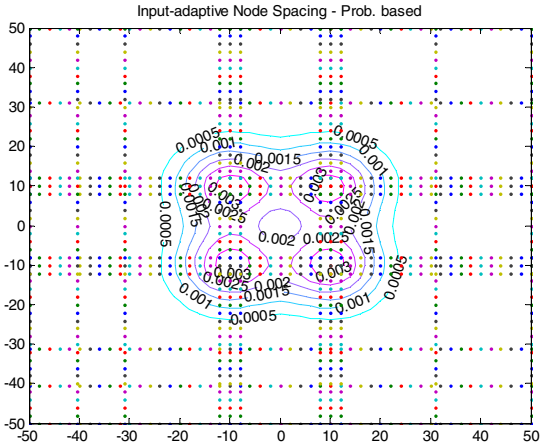
Table 1: Gain in dB over uniform node placement

$$J = 20 \log \frac{E_{uniform}}{E_{Adaptive}} \qquad (5)$$

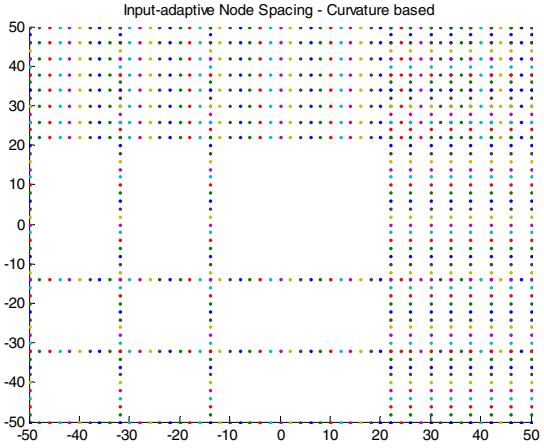
where $E_{uniform}$ and $E_{Adaptive}$ respectively refer to the error as in Eqn (1) for uniform vs. significance function adapted node placement. For each of the three cases visualized in Figs. 4(b) through (d), the corresponding gain is quantified in Table 2. A gain of 20 dB for example means that the error via adaptive node-spacing is about one-tenth of the error incurred with uniform node-spacing. (the logarithm is base 10). The success in approximating multi-dimensional transforms is hence readily apparent.



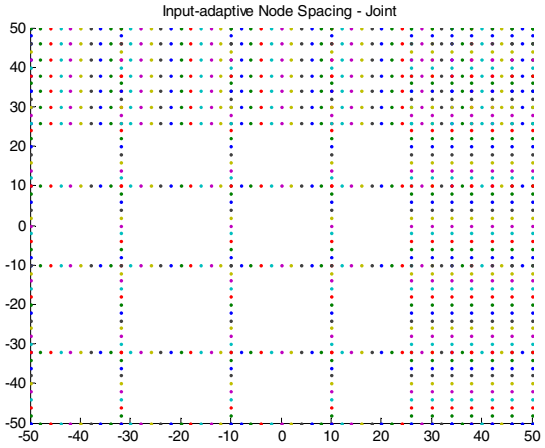
(a)



(b)



(c)



(d)

Fig. 4: Visualization of LUT node placement based on different criterion.

5. Conclusion

Designing regular lattices for look-up-tables that are desired to approximate complex non-linear multi-dimensional mappings poses a significant challenge. In all generality, for a pre-determined LUT size or equivalently the number of nodes, the optimal node selection problem is computationally intractable. This paper proposes an efficient, though sub-optimal algorithm to determine lattice nodes for color LUTs. The proposed method makes departures from previous efforts in two respects: 1.) while much existing work focuses exclusively on adapting nodes to curvature of the transform, the proposed sort-damp-select algorithm adapts to both the input distribution and transform curvature by unifying them into a single significance function, and 2.) the proposed method offers attractive scalability even as multi-dimensional nodes are selected in contrast to past techniques, which resort to separable node selection for efficiency. An exciting opportunity for future work includes providing local optimality guarantees in node selection under fixed interpolation models. Another key problem is the joint optimization of the node positions as well as the output value to be placed at the nodes.

6. References

1. R. Bala, “Device Characterization”, *Digital Color Imaging Handbook*, Chapter 5. CRC Press, 2003.
2. J. Z. Chang, J. P. Allebach, and C. A. Bouman, "Sequential Linear Interpolation of Multidimensional Functions," *IEEE Trans. on Image Processing*, Vol. 6, pp. 1231- 1245, September 1997.
3. M. J. Baines, “Algorithms for Optimal Discontinuous Piecewise Linear and Constant L_2 Fits to Continuous Functions”, *Mathematics of Computation,* vol. 62, pp. 645-669, April 1994.
4. *Vector Quantization and Signal Compression*, A. Gersho and R. Gray, Springer Verlag, 1990.

5. *Introduction to Graph Theory,* D. West, Prentice Hall, 2nd Edition, 2001.
6. G. Braudaway, "A procedure for optimum choice of a small number of colors from a large color palette for color imaging", *Proc. Electronic Imaging '86,* Boston, MA, Nov 86, pp75-79.

Author Biography

Vishal Monga received his B. Tech degree from Indian Institute of Technology (IIT), Guwahati in 2001 and his M.S. and PhD degrees in Electrical Engineering from The University of Texas at Austin in May 2003 and Aug 2005 respectively. His research interests lie broadly in statistical signal and image processing. He has researched problems in color imaging, multimedia security and statistical learning. In color imaging, his interests span a variety of processing blocks in the printer and digital camera pipeline including color halftoning by error diffusion, color device modeling, and non-separable color transforms.